



TITLE:

On Applying Scott's Logic to Termination Problems (アルゴリズムにおける証明論)

AUTHOR(S):

ANDO, SHIGERU; ITO, TAKAYASU

CITATION:

ANDO, SHIGERU ...[et al]. On Applying Scott's Logic to Termination Problems (アルゴリズムにおける証明論). 数理解析研究所講究録 1975, 236: 1-7

ISSUE DATE:

1975-05

URL:

<http://hdl.handle.net/2433/105512>

RIGHT:

On Applying Scott's Logic to Termination Problems

S. ANDO

T. ITO

(Mitsubishi Electric Corp.)

§ 1. Introduction

Scott's logic incorporates a very powerful induction rule called fixed-point induction which realizes various types of structural induction in a uniform manner. Termination problems are also nicely treated in Scott's system as is demonstrated by the typical examples given in §2, §3, where termination means termination under any computation rule which is a fixed-point one (see Manna 1972).

We propose the following method of presentation of termination problems. Let the domain of computation be such a primitive data type D as $D = S \cup \{UU\}$ $UU \notin S$

$$\forall x, y \in S. x \leq y \rightarrow x = y$$

$$\forall x \in S. UU \leq x$$

where 'S' is conceived to be the set of well-defined elements and 'UU' the undefinedness. Then, we introduce a recursively defined function ter from D to the data type of truth value TV such as

$$ter(UU) = UU$$

$$ter(x) = true \ (x \neq UU)$$

which is similar to the ' δ ' of Milner 1972 but the recursive definition allows us to exploit the well-founded structure of D which, as is shown by Kanayama 1972, is essential in termination proof.

Under these circumstances the termination of the computation which is represented by the recursive definition of the function f is expressed as

$$\lambda x. g(x) \subseteq \lambda x. ter(f(x))$$

where g is also a recursively-defined function which never takes value 'false', which reads

"Whenever condition g is satisfied, f terminates."

§ 2. Natural number

The domain of natural number is categorically specified by the following axioms \mathcal{N} (Scott 1969)

1. $\lambda x. ((x+1) - 1) \equiv \lambda x. x$
2. $\text{zero } (UU) \equiv UU$
3. $0 - 1 \equiv UU$
4. $\text{af. } \lambda x. \text{ if zero } (x) \text{ then } 0 \text{ wlse } f(x-1)+1 \equiv \lambda x. x$

where '+1' '-1', means the successor and the predecessor function respectively, and the predicate 'zero' means that the argument is zero.

Under these axioms, 'ter' is given as follows.

$\text{ter} \equiv \text{af. } \lambda x. \text{ if zero } (x) \text{ then true else } f(x-1)$

Example 1. Addition

$+ \equiv \text{af. } \lambda xy. \text{ if zero}(y) \text{ then } x \text{ else } f(x, y-1)+1$

Our goal is

$\mathcal{N} \vdash \text{ter}(x) \ \& \ \text{ter}(y) \subseteq \text{ter}(x+y)$

(where $\& \equiv \lambda xy. \text{ if } x \text{ then } y \text{ else false}$.)

it is easily obtained that

$x \subseteq \text{true}, y \subseteq \text{true} \vdash x \& y \subseteq y \ \wedge \ x \& y \subseteq x$

$x \subseteq y, x \subseteq z \vdash x \subseteq y \& z$

to which the induction rule is applied with respect to the 'ter' of $\text{ter}(y)$ generating the following two sub-goals.

1. $\mathcal{N} \vdash \text{ter}(x) \ \& \ UU \subseteq \text{ter}(x+y)$
2. $\mathcal{N}, \lambda ab. \text{ter}(a) \ \& \ \text{tl}(b) \subseteq \lambda ab. \text{ter}(a+b) \vdash \text{ter}(x) \ \& \ (\text{if zero}(y) \text{ then true else } \text{tl}(y-1)) \subseteq \text{ter}(x+y)$

1. is immediately verified. 2. is decomposed by the cases rule with respect to $\text{zero}(y)$, into the following.

- 2.1 $\mathcal{N}, \lambda ab. \text{ter}(a) \ \& \ \text{tl}(b) \subseteq \lambda ab. \text{ter}(a+b), \text{zero}(y) \equiv \text{true} \vdash \text{ter}(x) \ \& \ \text{true} \subseteq \text{ter}(x)$

which is immediately verified

2.2 $\forall, \lambda ab. \text{ter}(a) \& \text{tl}(b) \subseteq \lambda ab. \text{ter}(a+b), \text{zero}(y) \equiv \text{false}$
 $\vdash \text{ter}(x) \& \text{tl}(y-1) \subseteq \text{ter}(x+y)$

which we can prove by letting a and b of the assumption x and y-1 respectively and using modus-ponens rule and transitivity axiom involving

$$\text{zero}(y) \equiv \text{false} \vdash \text{ter}(x+y) \equiv \text{ter}((x+(y-1))+1) \equiv \text{ter}(x+(y-1))$$

Example 2. Fibonacci function

$$\text{Fib}(x) \equiv \text{if zero}(x) \text{ then } 1 \text{ else if } x=1 \text{ then } 1 \text{ else} \\ \text{Fib}(x-1) + \text{Fib}(x-2)$$

(where $x=1$ is an abbreviation of $\text{zero}(x-1)$ and $x-2, (x-1)-1$.)

We presuppose the termination of + i.e. $\text{ter}(x) \& \text{ter}(y) \subseteq \text{ter}(x+y)$

Let $P \equiv \lambda x. \text{ter}(\text{Fib}(x))$ then,

$$\text{if zero}(x) \text{ then true else if } (x=1) \text{ then true else } p(x-1) \& p(x-2) \subseteq p(x) \quad \dots 1)$$

Our goal being $\text{ter}(x) \subseteq p(x)$

subgoal $\text{ter}(x) \subseteq p(x) \& p(x+1)$ is chosen for the reason of induction.

The induction subgoals are

1. $\vdash \text{UU}(x) \subseteq p(x) \& p(x+1)$ which is immediate.
2. $\text{tl} \subseteq \lambda a. p(a) \& p(a+1)$

$$\vdash \text{if zero}(x) \text{ then true else } \text{tl}(x-1) \subseteq p(x) \& p(x+1)$$

Subgoal 2 is decomposed by the cases rule into the following.

- 2.1 $\vdash \text{true} \subseteq p(0) \& p(1)$ (case $\text{zero}(x) \equiv \text{true}$)
- 2.2 $\vdash \text{tl}(0) \subseteq p(1) \& p(2)$ (case $x=1 \equiv \text{true}$)
- 2.3 $\text{zero}(x) \equiv (x=1) \equiv \text{false}, \text{tl} \subseteq \lambda a. p(a) \& p(a+1)$
 $\vdash \text{tl}(x-1) \subseteq p(x) \& p(x+1)$

2.1 and 2.2 is verified by proving

$$p(0) \equiv p(1) \equiv p(2) \equiv \text{true}$$

To prove 2, 3, first we obtain

$$\text{tl}(x-1) \subseteq p(x-1) \& p(x)$$

by applying x-1 to the assumption,

$$\text{tl}(x-1) \subseteq p(x+1)$$

is obtained from 1) and $\text{zero}(x) \equiv (x=1) \equiv \text{false}$, therefore we obtain, with definition of &,

$$\text{tl}(x-1) \subseteq p(x-1) \& p(x) \& p(x+1) \subseteq p(x) \& p(x+1) \quad \text{Q.E.D}$$

Example 3. Ackermans function

$Ack \equiv \lambda x. \lambda y. \text{if zero}(x) \text{ then } y+1 \text{ else if zero}(y) \text{ then } f(x-1, 1)$
 $\text{else } f(x-1, f(x, y-1))$

Let $p(x, y) \equiv \text{ter}(Ack(x, y))$ then

$p(x, y) \equiv \text{if zero}(x) \text{ then } \text{ter}(y) \text{ else if zero}(y) \text{ then } p(x-1, 1)$
 $\text{else } p(x-1, Ack(x, y-1))$

Our goal is

$\vdash \text{ter}(x) \ \& \ \text{ter}(y) \sqsubseteq p(x, y)$

First we resort to the induction rule with respect to the 'ter' of $\text{ter}(x)$ which generates the following two subgoals

1. $\vdash \text{UU}(x) \ \& \ \text{ter}(y) \sqsubseteq p(x, y)$

2. $\lambda ab. \text{tl}(a) \ \& \ \text{ter}(b) \sqsubseteq p$

$\vdash (\text{if zero}(x) \text{ then true else } \text{tl}(x-1)) \ \& \ \text{ter}(y) \sqsubseteq p(x, y)$

1. is verified immediately

2. is decomposed by the cases rule with respect to $\text{zero}(x)$ into

2.1. $\text{zero}(x) \equiv \text{true} \vdash \text{true} \ \& \ \text{ter}(y) \sqsubseteq p(0, y) \equiv \text{ter}(y)$

which is immediately verified, and

2.2 $\text{tl}(a) \ \& \ \text{ter}(b) \sqsubseteq p(a, b); \text{zero}(x) \equiv \text{false} \vdash \text{tl}(x-1) \ \& \ \text{ter}(y)$
 $\sqsubseteq p(x, y)$

From 2.2 the induction rule with respect to the 'ter' of $\text{ter}(y)$ generates the following.

2.2.1 $\text{tl}(x-1) \ \& \ \text{UU}(y) \sqsubseteq p(x, y)$

2.2.2 $\lambda ab. \text{tl}(a) \ \& \ \text{ter}(b) \sqsubseteq p, \text{zero}(x) \equiv \text{false}, \lambda cd. \text{tl}(c-1) \ \& \ \text{t2}(d) \sqsubseteq p$

$\vdash \text{tl}(x-1) \ \& \ (\text{if zero}(y) \text{ then true else } \text{tl}(y-1)) \sqsubseteq p(x, y)$

2.2.1 is immediate, 2.2.2 is decomposed by the cases rule with respect to $\text{zero}(y)$ into the following 2.2.2.1 and 2.2.2.2.

2.2.2.1

$\text{zero}(y) \equiv \text{true}, \text{zero}(x) \equiv \text{false} \lambda ab. \text{tl}(a) \ \& \ \text{ter}(b) \sqsubseteq p$
 $\text{tl}(x-1) \ \& \ \text{true} \sqsubseteq p(x, y)$

PROOF $\lambda a \lambda b. \text{tl}(a) \ \& \ \text{ter}(b) \sqsubseteq p \vdash \text{tl}(x-1) \ \& \ \text{ter}(1) \sqsubseteq p(x-1, 1)$

$\text{zero}(x) \equiv \text{false}, \text{zero}(y) \equiv \text{true} \vdash p(x-1, 1) \equiv p(x, y)$

By applying modus-ponens and transitivity several times to these,
we can obtain 2.2.2.1

2.2.2.2

$\text{zero}(y) \equiv \text{zero}(x) \equiv \text{false}, \lambda ab. t1(a) \ \& \ \text{ter}(b) \sqsubseteq p, \lambda cd. t1(c-1) \ \& \ t2(d) \sqsubseteq p$

$\vdash t1(x-1) \ \& \ t2(y-1) \sqsubseteq p(x \ y)$

PROOF

From

$\lambda ab. t1(a) \ \& \ \text{ter}(b) \sqsubseteq p \vdash t1(x-1) \ \& \ p(x, y-1) \sqsubseteq p(x-1, \text{ack}(x, y-1))$ and
 $\text{zero}(x) \equiv \text{zero}(y) \equiv \text{false} \vdash p(x-1, \text{ack}(x, y-1)) \sqsubseteq p(x \ y)$ several applications
of modus ponens and transitivity gives

assumptions $\vdash t1(x-1) \ \& \ p(x, y-1) \sqsubseteq p(x \ y)$

And several application of modus ponens and transitivity to this
and

$\lambda cd. t1(c-1) \ \& \ t2(d) \sqsubseteq p \vdash t1(x-1) \ \& \ t2(y-1) \sqsubseteq p(x, y-1)$

and the definition of '&' gives

assumptions $\vdash t1(x-1) \ \& \ t2(y-1) \sqsubseteq p(x \ y) \quad \text{Q. E. D}$

§ 3. List

The axioms of list is similar to that of natural number

1. $\lambda x \lambda y. \text{car}(\text{cons}(x \ y)) \equiv \lambda x \lambda y. x$
2. $\lambda x \lambda y. \text{cdr}(\text{cons}(x \ y)) \equiv \lambda x \lambda y. y$
3. if $\text{atom}(x)$ then $\text{car}(x)$ else $UU \equiv UU$
4. if $\text{atom}(x)$ then $\text{cdr}(x)$ else $UU \equiv UU$
5. $\text{atom}(\text{NIL}) \equiv \text{true}$
6. $\alpha f. \lambda x. (\text{if } \text{null}(x) \text{ then NIL}$
 else if $\text{atom}(x)$ then x
 else $\text{cons}(f(\text{car}(x)) f(\text{cdr}(x))) \equiv \lambda x. x$

In this case the definition of ter is

$\text{ter} \equiv \alpha f. \lambda x. \text{if } \text{atom}(x) \text{ then true}$
 else $f(\text{car}(x)) \ \& \ f(\text{cdr}(x))$

Example Primitive Recursive Function of lists

$f1(x, y1, \dots, yk)$

\equiv if atom(x) then $g2(x, y1, \dots, yk)$

else $h1(x, f1(car(x), y1, \dots, yk), f2(cdr(x), y1, \dots, yk),$
 $y1, \dots, yk)$

$f2(x, y1, \dots, yk)$

\equiv if atom(x) then $g2(x, y1, \dots, yk)$

else $h2(x, f1(car(x), y1, \dots, yk), f2(cdr(x), y1, \dots, yk),$
 $y1, \dots, yk)$

From the above definition we can prove the termination of $f1$ and $f2$ under the assumption that $g1$, $g2$, $h1$, and $h2$ terminates.

§ 4. Conclusion

Termination of a computation is always equivalent to downward monotonicity of that computation process with respect to some well-founded relation (Y. Kanayama 1972). That is why structural induction is essential in termination problems. This feature is also explicit in Floyd-Manna's first-order method because it always resorts to some axiom scheme of structural induction, say, mathematical induction in the domain of natural number. The success of the proof depends totally upon the choice of the well-founded relation, which is not at all uniform and relies heavily upon intuition. On the contrary, in our method we may not be conscious of the well-founded structure explicitly. It is implicitly incorporated in the recursive definitions of conditions on the arguments. Fixed-point induction with respect to that condition function automatically generates subgoals which reflects the well-founded structure that is needed to establish the termination. Of course, in case very complicated well-founded relation is involved, some ingenuity is necessary in designing the condition function. But in any case, our formal frame-work is simple enough to make the whole reasoning transparent, and helps to

guide our intuition. It is a good candidate for the formalism to be adopted when trying to mechanize the termination problem.

References

- Kleene 1952 Introduction to Metamathematics. North Holland
- Scott 1969 A Type-theoretical alternative to Unpublished
- Milner 1972 "Implimentation and -" proc. of A.C.M
conf. New Mexico
- Manna 1972 "Inductive methods - " *the same as above*
- Kanayama 1972 "Algebraic Properties of Program"
proc of 1st Japan U.S.A computer conf.